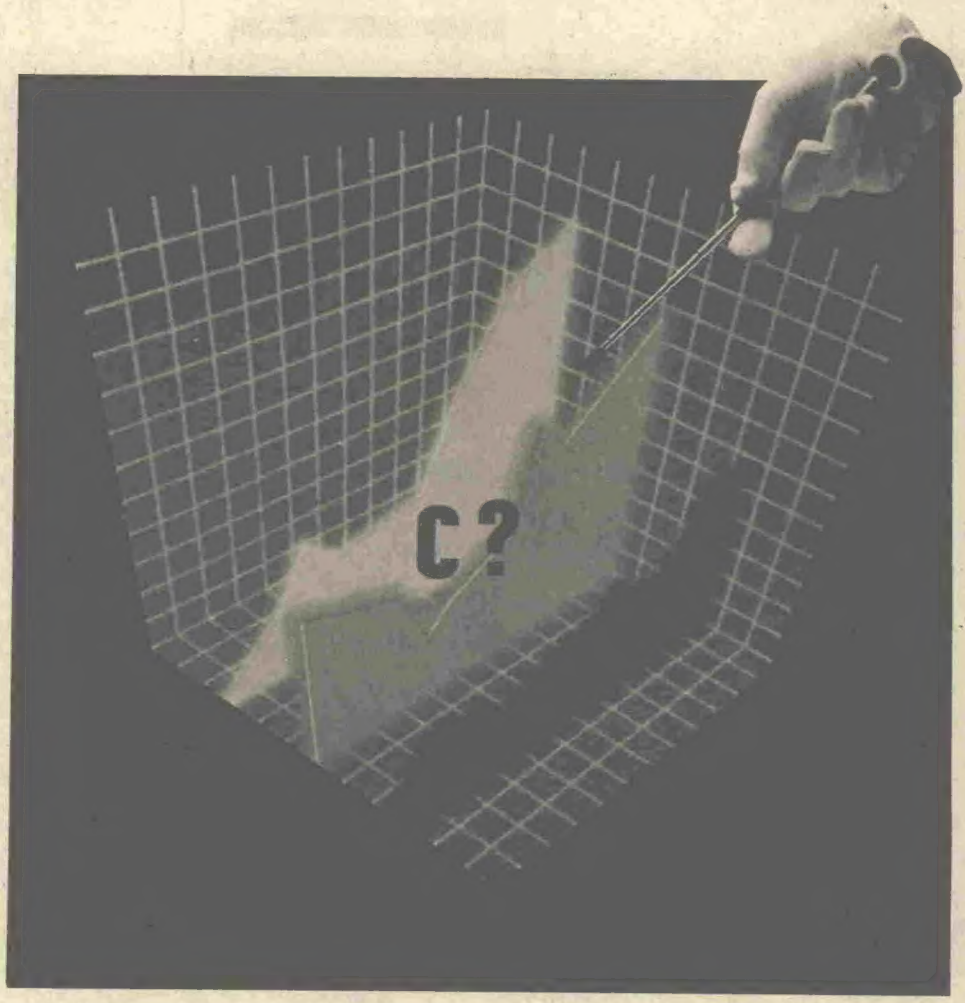
กระเทาะเปลือกภาษาซี (1) : 14 คำถาม

ไขปัญหาสารพัด ให้ถึงแก่นกับ 14 คำถามภาษาซี



ในช่วงสี่ห้าปีที่ผ่านมานี้ ดูเหมือนว่าภาษาซีได้มีความสำคัญในการพัฒนาโปรแกรมบนเครื่องระดับไมโครคอมพิวเตอร์มากขึ้น จนกล่าวได้ว่าเป็นภาษาสุดฮิตของผู้ผลิตซอฟต์แวร์ทั้งหลาย และปัจจุบันนี้ไมโครซอฟต์วินโดว์ซึ่งเป็นระบบวินโดว์บนดอสได้รับความนิยมอย่างสูง ภาษาซีก็ยิ่งมีบทบาทมากขึ้น เพราะการพัฒนาโปรแกรมบนวินโดว์นั้น ผู้พัฒนามักจะเลือกใช้ภาษาซีเป็นหลัก สาเหตุที่ไม่ใช้ภาษาอื่นเนื่องจากยุ่งยากและไม่สะดวกต่อการพัฒนา อาจจะมีการใช้ C++ บ้าง แต่ C++ ก็มีพื้นฐานมาจากภาษาซี จากที่กล่าวมานั้นเป็นการแสดงให้เห็นบทบาทและความสำคัญของภาษาซี ดังนั้นบทความนี้จะนำผู้อ่านไปพบกับความลึกซึ้งและซับซ้อนของภาษาซีเพื่อให้เข้าใจและนำไปใช้งานต่อไป

บทความนี้มีลักษณะเป็นการถามตอบเกี่ยวกับปัญหาที่พบในการใช้งานภาษาซี คำถามแต่ละคำถามจะมีความยากง่ายแตกต่างกันไป สำหรับประเภทของคำถามจะแบ่งออกเป็นหัวข้อดังนี้

1. Null Pointers
2. Arrays and Pointers
3. Order of Evaluation
4. ANSI C
5. C Preprocessor
6. Variable-Length Argument Lists
7. Memory Allocation
8. Structures
9. Declarations
10. Boolean Expressions and Variables
11. Operating System and Dependencies
12. Stdio
13. อื่น ๆ

ในตอนที่ 1 นี้จะทำการตอบคำถามเกี่ยวกับ Null Pointers เป็นส่วนใหญ่ สำหรับในข้ออื่น ๆ อาจจะมีแทรกในบางส่วนสำหรับการตอบคำถาม

1. null pointer คืออะไร มีความสำคัญอย่างไร

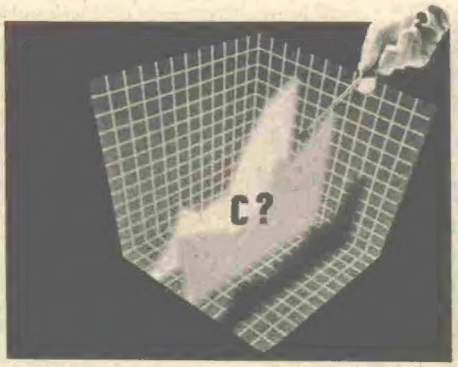
ตอบ ตามคำจำกัดความของภาษาแล้ว พอยน์เตอร์ทุกชนิดจะมีค่าพอยน์เตอร์พิเศษค่าหนึ่งเรียกว่า "null pointer" ค่าพอยน์เตอร์นี้จะแตกต่างจากค่าพอยน์เตอร์อื่น ๆ ตรงที่ไม่ใช่แอดเดรสของออปเจ็กต์ใด ๆ นั่นก็คือโอเปอร์เรเตอร์จะไม่ใช้ค่า null pointer เลย เพราะโอเปอร์เรเตอร์นี้จะให้ค่าแอดเดรสของออปเจ็กต์เสมอ ส่วนค่าที่รีเทอร์นจากฟังก์ชัน malloc นั้นจะเป็น null pointer เมื่อไม่สามารถ allocate ที่ได้ซึ่งหมายความว่าไม่ได้ชี้ไปที่ออปเจ็กต์ใด ๆ

null pointer แตกต่างจากพอยน์เตอร์ที่ยังไม่ได้กำหนดค่า (uninitialized pointer) โดยสิ้นเชิงคือไม่ได้ชี้ที่ออปเจ็กต์อะไรเลย ส่วนพอยน์เตอร์ที่ยังไม่ได้กำหนดค่าจะชี้ที่ตำแหน่งใดก็ได้ เช่น ชี้ไปที่ยังไม่ได้ allocate หรือชี้ไปที่ออปเจ็กต์ใด ๆ ในโปรแกรมหรือชี้ไปที่ "ขยะ"

ตามที่ได้กล่าวมาข้างต้น พอยน์เตอร์แต่ละชนิดจะมี null pointer สำหรับแต่ละชนิดและค่าจริงของ null pointer (ค่าขณะรันโปรแกรม) อาจจะแตกต่างกันตามปกติแล้วโปรแกรมเมอร์ไม่จำเป็นต้องรู้ค่าจริงของ null pointer แต่สำหรับผู้เขียนคอมพายเลอร์จำเป็นต้องรู้เพื่อที่จะใช้ null pointer ของแต่ละชนิดให้ตรงกับพอยน์เตอร์ชนิดนั้น ๆ

2. เวลาเขียนโปรแกรม จะเขียน null pointer อย่างไร

ตอบ ตามคำจำกัดความของภาษาแล้ว ค่าคงที่ศูนย์ (0) ที่อยู่ใน pointer context จะถูกแปลงให้เป็น null pointer ในช่วงเวลาคอมพายล์ คือในประโยค



initialization, assignment หรือ comparison เมื่อมีข้างหนึ่งเป็นตัวแปรพอยน์เตอร์หรือนิพจน์พอยน์เตอร์ คอมพายเลอร์จะรับรู้ว่าค่าคงที่ศูนย์ในอีกข้างหนึ่งเป็น null pointer และเลือกค่าจริงของ null pointer ให้ตรงกัน ดังนั้นประโยคต่อไปนี้จึงถูกต้องโดยสมบูรณ์

```
char *p = 0;
if (p != 0)
```

อย่างไรก็ตาม มีกรณีที่คอมพายเลอร์ไม่สามารถตรวจสอบว่า ค่าคงที่ศูนย์นั้นเป็น null pointer ในระบบยูนิกซ์มีคำสั่ง "execl" ซึ่งรับพารามิเตอร์ไม่จำกัด ในการส่งพารามิเตอร์ไปให้ "execl" ต้องปิดท้ายด้วย null pointer เสมอเช่น

```
execl ("/bin/sh", "sh",
"-c", "ls", (char*) 0 ) ;
```

ถ้าไม่มีการ cast ด้วย (char*) แล้ว คอมพายเลอร์จะไม่สามารถรู้ได้ว่าเป็นการผ่านค่า null pointer และจะผ่านค่า integer 0 แทน เมื่อมีการใช้ function prototype การผ่านค่าไปให้ฟังก์ชันจะเสมือนเป็นการ assignment ไม่จำเป็นต้อง cast เพราะ prototype กำหนดชนิดของพารามิเตอร์ไว้แล้ว ตัวคอมพายเลอร์จะทำการตรวจสอบว่าจะต้องแปลงค่าคงที่ศูนย์ให้เป็น null pointer หรือไม่ สำหรับฟังก์ชันรับพารามิเตอร์ไม่จำกัด เช่น printf หรือ execl แล้ว prototype ไม่สามารถบอกชนิดของพารามิเตอร์ได้ ดังนั้นจะเป็นการปลอดภัย ในการ cast พารามิเตอร์ให้เป็น null pointer เมื่อมีการเรียกใช้ฟังก์ชันแบบพารามิเตอร์ไม่จำกัดหรือฟังก์ชันที่ไม่มี prototype ตารางที่ 1 แสดงการใช้ค่าต่าง ๆ

3. ตัวแปรพอยน์เตอร์เหมือนกับตัวแปร int หรือไม่

ใช้ค่าคงที่ศูนย์ แบบไม่ต้อง cast	ใช้ค่าคงที่ศูนย์ แบบ cast
– initialization – assignments – comparisons – เรียกใช้ฟังก์ชันที่มี prototype และมีจำนวนพารามิเตอร์แน่นอน	– เรียกใช้ฟังก์ชันไม่มี prototype – เรียกใช้ฟังก์ชันที่รับพารามิเตอร์ไม่จำกัด

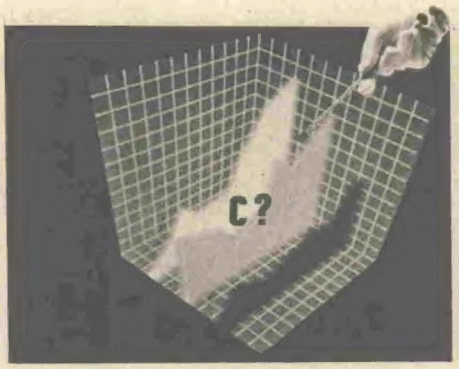
ตารางที่ 1

ตอบ ไม่เหมือนกัน เพราะการแปลงพอยน์เตอร์ให้เป็นอินทีเจอร์หรือแปลงอินทีเจอร์ให้เป็นพอยน์เตอร์ไม่ควรกระทำเป็นอย่างยิ่ง เพราะจะทำให้โปรแกรมไม่ portable (สำหรับพอยน์เตอร์นั้นสามารถ cast ให้เป็นพอยน์เตอร์ "universal" ได้ เมื่อต้องการผ่านพอยน์เตอร์หลาย ๆ ชนิด โดยที่ void* เป็นพอยน์เตอร์ "universal" สำหรับ ANSI compiler ส่วน char* เป็นพอยน์เตอร์ "universal" สำหรับคอมพายเลอร์ที่มีมาก่อน ANSI)

4. แล้ว NULL คืออะไร และถูกกำหนดไว้อย่างไร

ตอบ ปกติแล้วโปรแกรมเมอร์มักจะไม่ต้องการให้โปรแกรมของตนเต็มไปด้วย null pointer ที่เป็นศูนย์ ด้วยเหตุนี้เอง NULL จึงถูกกำหนดให้เป็น preprocessor macro ซึ่งมีค่าเป็น 0 (หรือ (void*) 0 ซึ่งจะกล่าวถึงต่อไป) ซึ่งอาจจะกำหนดไว้ใน <stdio.h> หรือ <stddef.h> ดังนั้นโปรแกรมเมอร์จึงไม่สับสนว่าเลขศูนย์นั้นเป็น integer 0 หรือ null pointer เพราะจะใช้ NULL แทนเลขศูนย์เมื่อมีการอ้างถึง null pointer อย่างไรก็ตาม NULL จะถูกแทนที่ด้วยเลขศูนย์โดย preprocessor ก่อนที่จะส่งไปคอมพายเลอร์ ซึ่งทำให้ต้อง cast อยู่หน้า NULL เมื่อมีการเรียกใช้ฟังก์ชันแบบรับพารามิเตอร์ไม่จำกัดหรือฟังก์ชันที่ไม่มี prototype ในตารางที่ 1 สามารถใช้กับ NULL ได้

NULL ควรจะใช้กับพอยน์เตอร์เท่านั้น และไม่ควรใช้ NULL ในลักษณะที่เป็นเลขศูนย์ของข้อมูลชนิดอื่น ถึงแม้ว่ามันจะทำงานได้ก็ตาม เพราะผิดกฎที่กำหนดไว้และอาจก่อให้เกิดความสับสน



ได้ง่าย (สำหรับ ANSI จะกำหนดให้เป็น NULL มีค่าเป็น (void*) 0 ซึ่งทำให้ไม่สามารถใช้ NULL กับข้อมูลชนิดอื่นได้) อีกกรณีหนึ่งคือ ห้ามใช้ NULL แทน ASCII null character ควรจะกำหนด null character ขึ้นมาใหม่เช่น

```
# define NUL '\0'
```

5. บนเครื่องทั่ว ๆ ไป ค่าจริงของ null pointer มักจะมีค่าเป็นศูนย์ทุกบิตและจะกำหนดให้ NULL เป็นเลขศูนย์ แต่บนเครื่องที่ค่าจริงของ null pointer ประกอบด้วยบิตที่ไม่เป็นศูนย์ทั้งหมด ในกรณีนี้เราจะกำหนด NULL อย่างไร

ตอบ โปรแกรมเมอร์ไม่จำเป็นต้องรู้ค่าจริงของ null pointer ซึ่งอยู่ภายในเครื่องเพราะคอมพายเลอร์จะคอยดูแลในส่วนนี้เอง ถ้ามีการใช้ค่าจริงของ null pointer ที่ประกอบด้วยบิตที่ไม่เป็นศูนย์คอมพายเลอร์จะรับผิดชอบในการแปลง null pointer (ซึ่งถูกเขียนแทนด้วย 0 หรือ NULL) ให้เป็นไปตามค่าจริงของ null pointer นั้น ดังนั้นการกำหนดให้ NULL มีค่าเป็นเลขศูนย์จึงสามารถใช้กับทุก ๆ เครื่องไม่ว่าเครื่องนั้นจะมีค่าจริงของ null pointer เป็นอย่างไร

6. ถ้ากำหนด NULL ในลักษณะต่อไปนี้

```
#define NULL (char*) 0
```

การผ่านค่าไปให้ฟังก์ชัน โดยไม่ต้อง cast จะทำให้โปรแกรมทำงานได้ตามปกติหรือไม่

ตอบ ไม่เสมอไป ปัญหาคือว่า มีบางเครื่องที่ค่าจริงของ null pointer ของแต่ละชนิดมีค่าต่างกัน ถ้าฟังก์ชันรับพารามิเตอร์แบบ char* ก็จะไม่มีปัญหา แต่ถ้าเป็นพอยน์เตอร์ชนิดอื่น จะทำให้เกิดปัญหาได้เช่น

```
FILE *fp = NULL,
```

อย่างไรก็ตาม ANSI ได้กำหนดค่าของ NULL ได้ดังนี้

```
#define NULL (void*) 0
```

การกำหนดค่าในลักษณะนี้จะช่วยแก้ปัญหาดังกล่าวได้ (เฉพาะบนเครื่องที่มีพอยน์เตอร์ทุกชนิดเหมือนกัน) นอกจากนี้ยังแก้ปัญหาการใช้ NULL ที่ผิด ๆ ได้ด้วย (เช่นใช้ NULL เป็น null character หรือเป็น integer 0)

7. การใช้ macro ดังนี้

```
#define Nullptr (type)
(type*) 0
```

จะช่วยให้ null pointer ตรงกับชนิดข้อมูลหรือไม่

ตอบ เทคนิคนี้มักจะนิยมใช้ในกลุ่มของโปรแกรมเมอร์มือใหม่ การกำหนดในลักษณะนี้ไม่จำเป็นสำหรับประโยค assignment และ comparison (ดูจากคำถามที่ 2) นอกจากจะทำให้โปรแกรมยาวแล้วยังแสดงให้เห็นว่าโปรแกรม-

เมอร์ยังไม่มีความเข้าใจในเรื่อง null pointer ดีพอ ข้อเสียอื่น ๆ เช่น ทำให้ต้องกลับไปอ่านความหมายของมาโครตัวนี้ใหม่ และการใช้งานต้องใช้ด้วยความระมัดระวังเป็นอย่างสูง

8. การเปรียบเทียบแบบย่อ "if (p) เพื่อทดสอบว่า p เป็น null pointer หรือไม่ จะเกิดอะไรขึ้นถ้าค่าจริงของ null pointer ประกอบด้วยบิตที่ไม่เป็นศูนย์

ตอบ ในภาษาซีค่าบูลลีนของนิพจน์จะถูกใช้ในประโยค if, while, for, do statement และใช้กับเครื่องหมาย &&, ||, !, ? : ค่าบูลลีนของนิพจน์ มี 2 ค่า คือ ค่าเท็จและค่าจริง ค่าเท็จเกิดขึ้นเมื่อนิพจน์เป็นศูนย์ นอกจากนั้นแล้วจะได้ค่าจริงดังนี้

```
if (expr)
```

โดย expr เป็นนิพจน์ใด ๆ จึงมีค่าเท่ากับ

```
if (expr! = 0)
```

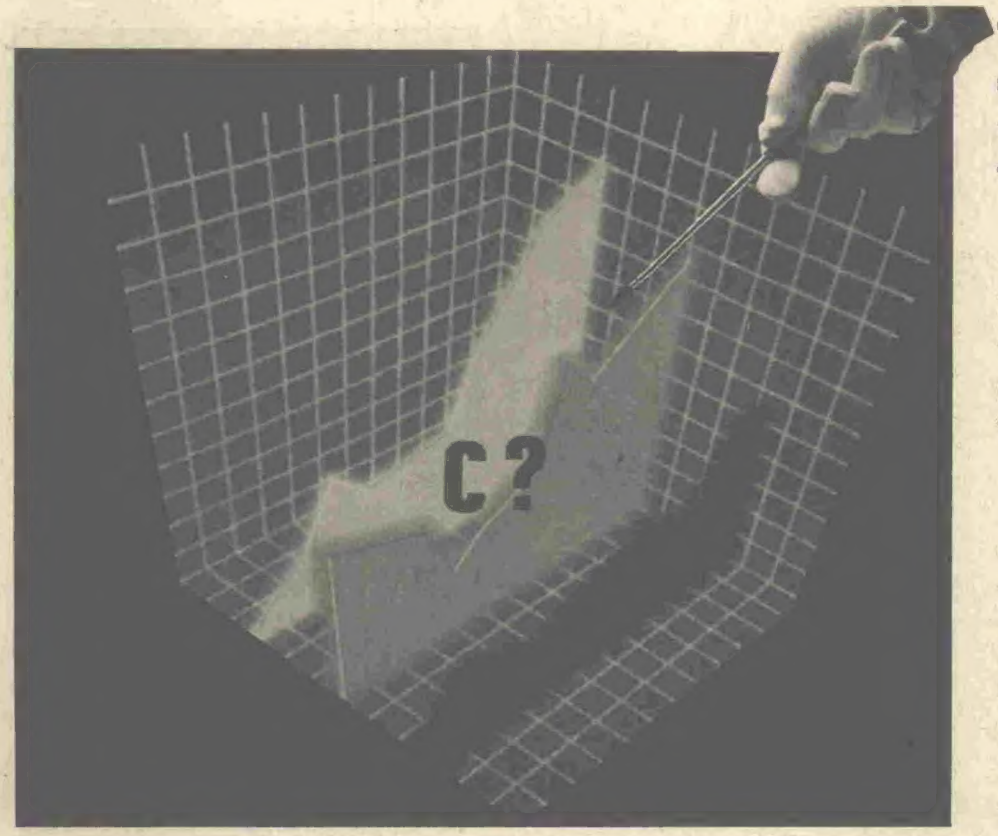
โดยการแทน expr ด้วย p จะได้

```
if (p) มีค่าเท่ากับ if (p ! = 0)
```

นี่คือประโยคเปรียบเทียบ ดังนั้นคอมพายเลอร์จะรู้ได้ว่าเลขศูนย์นี้คือ null pointer ดังนั้น if (p) จึงเป็นประโยคที่ถูกต้องโดยสมบูรณ์ และ if (p) กับ if (p ! =0) จะให้โค้ดที่เหมือนกัน ค่าจริงของ null pointer ไม่เกี่ยวข้องกันหรือมีผลต่อกรณีนี้เลย

สำหรับเครื่องหมาย ! สามารถอธิบายได้ดังนี้

```
!expr มีค่าเท่ากับ expr? 0 : 1
ผู้อ่านลองไปคิดเป็นการบ้านว่า
if (!p) มีค่าเท่ากับ if (p==0) ได้อย่างไร
```



9. ถ้า "NULL" มีค่าเท่ากับ "0" แล้วควรจะให้ค่าไหนดี

ตอบ โปรแกรมเมอร์จำนวนมากเชื่อว่า ควรใช้ "NULL" ในทุก ๆ ที่ที่เกี่ยวกับพอยน์เตอร์มากกว่า เพราะเป็นเครื่องเตือนว่าขณะนี้กำลังยุ่งเกี่ยวกับพอยน์เตอร์ อย่างไรก็ตามมีคนจำนวนหนึ่งรู้สึกว่าการใช้ NULL ทำให้เกิดความสับสน ดังนั้นจึงเลือกใช้ "0" แทนที่จะใช้ "NULL" คำตอบของคำถามนี้คือ ขึ้นอยู่กับความพอใจที่จะเลือกใช้ แต่โปรแกรมเมอร์ภาษาซีต้องเข้าใจว่า "NULL" และ "0" ใช้แทนซึ่งกันและกันได้ และการใช้ "0" โดยไม่ต้อง cast ก็เป็นที่ยอมรับโดยทั่วไป สำหรับ "NULL" นั้นไม่ควรวางใจว่าจะสามารถแยกความแตกต่างระหว่างพอยน์เตอร์ 0 กับ integer 0 ได้เสมอไป (ทั้งในแง่ความเข้าใจของโปรแกรมเมอร์หรือในแง่ของผู้เขียนคอมพายเลอร์) สุดท้ายนี้ไม่ควรใช้ NULL กับข้อมูลชนิดอื่นที่ไม่ใช้พอยน์เตอร์อีกด้วย

10. ในกรณีที่มีการเปลี่ยนแปลงค่าของ NULL (โดยเฉพาะบนเครื่องที่ค่าจริงของ null pointer ประกอบด้วยบิตที่ไม่เป็นศูนย์) จะไม่เป็นการดีกว่าหรือที่จะใช้ NULL แทน 0

ตอบ ไม่เลย ปกติเราจะใช้มาโครเพื่อจะทำการเปลี่ยนแปลงค่าของมาโคร สามารถกำหนด macro NULL ขึ้นมาเพื่อเตือนว่ากำลังทำงานเกี่ยวกับพอยน์เตอร์ ตัวภาษาเองก็รับประกันอยู่แล้วว่าเลขศูนย์ซึ่งอยู่ใน pointer context ถือว่าเป็น null pointer ส่วน NULL

เป็นทางเลือกทางหนึ่งที่ใช้ในการอ้างถึง null pointer เท่านั้น

11. มีคอมพายเลอร์ซึ่งจะต้องใช้ NULL เท่านั้น จึงจะคอมพายล์ผ่านได้หมายความว่าอย่างไร

ตอบ แสดงว่าคอมพายเลอร์นั้นใช้ไม่ได้

12. NULL คือศูนย์ ส่วน null pointer ไม่ใช่เลขศูนย์แล้ว NULL กับ null pointer แตกต่างกันอย่างไร

ตอบ "null pointer" คือ ตัวกำหนดของภาษาซึ่งค่าจริงจะเป็นอย่างไรก็ได้ (ขึ้นกับแต่ละเครื่อง บางเครื่องเป็นศูนย์ทุกบิต บางเครื่องไม่ใช่) เวลาเขียนโปรแกรมเราอ้างถึง "null pointer" ด้วยเลขศูนย์ "0" ส่วน "NULL" คือมาโครซึ่งถูกกำหนดให้เป็นเลขศูนย์ 0 หรือ (void*) 0

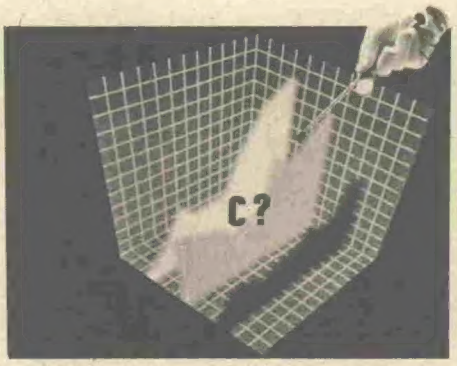
"null" และ "NULL" ถูกใช้อ้างถึงหลาย ๆ ความหมายตามแต่จะนิยมใช้กันจนมีความหมายหลายอย่างดังนี้

1. null pointer ตามที่กำหนดไว้ของภาษามีความหมายตามคำถามที่ 1

2. ค่าจริงของ null pointer ซึ่งอาจประกอบด้วยศูนย์ทุกบิตหรือไม่ก็ได้ และอาจจะแตกต่างกันตามชนิดของพอยน์เตอร์ ค่านี้ผู้เขียนคอมพายเลอร์จะเป็นผู้กำหนดขึ้นเอง โปรแกรมเมอร์ไม่ต้องสนใจ

3. ตัวแทนของ null pointer ในโปรแกรมภาษาซี คือ "0"

4. NULL macro ซึ่งถูกกำหนดเป็น "0" หรือ "(void) 0"



5. ASCII null character (NUL) ทุกบิตเป็นศูนย์หมดแต่ไม่เกี่ยวกับ null pointer

สำหรับบทความนี้ "null pointer" มีความหมายตามข้อ 1 เลขศูนย์ "0" มีความหมายตามข้อ 3 และ NULL มีความหมายตามข้อ 4

13. ทำไมจึงเกิดความสับสนเรื่อง null pointer มากเหลือเกิน

ตอบ มีหลายเหตุผลด้วยกันที่ทำให้เกิดความสับสนดังกล่าว เช่น

- โปรแกรมเมอร์ภาษาซี มักจะอยากรู้มากกว่าที่จำเป็นต้องรู้
- บนเครื่องทั่ว ๆ ไป null pointer ในไฟล์ต้นฉบับและค่าจริงภายในเครื่องจะเป็นศูนย์เหมือนกัน
- การใช้ macro NULL ทำให้คิดว่าจะมีการเปลี่ยนแปลงค่าของ NULL ในภายหลัง
- "0" มีความหมายตามที่ได้อธิบายไว้แล้ว ซึ่งอาจจะเป็น integer 0 หรือพอยน์เตอร์ 0 ก็ได้
- "null" ถูกใช้ในหลายความหมายจนเกินไป

วิธีป้องกันความสับสนก็คือ ให้คิดว่าภาษาซีมีคีย์เวิร์ดสำหรับ null pointer (เหมือนกับ NIL ในภาษาปาสคาล) และคีย์เวิร์ดนั้นก็คือ เลขศูนย์ "0"

14. สรุปแล้วจะใช้ null pointer ได้อย่างไร

ตอบ มีกฎง่าย ๆ 2 ข้อ คือ

1. เมื่ออ้างอิง null pointer ในไฟล์ต้นฉบับให้ใช้ "NULL" หรือ "0"

2. เมื่อมีการเรียกใช้ฟังก์ชันและส่ง "0" หรือ "NULL" ไปให้ฟังก์ชันให้ cast ให้เป็นชนิดตามต้องการเสียก่อน

ส่งท้าย

จากคำถามทั้ง 14 ข้อที่ได้ตอบไปแล้ว ถ้าเข้าใจในข้อ 1, 2, 4, 9, 13 แสดงว่าคุณสามารถที่จะเข้าใจเกี่ยวกับภาษาซีบ้างแล้ว สำหรับบางข้อที่ยังไม่ได้กล่าวถึงหรือตอบคำถามในตอนที่ 1 ผู้เขียนจะขอตอบคำถามในตอนต่อไปครับ กรุณาอดใจรอหน่อยนะครับ ขอให้โชคดี!

หนังสืออ้างอิง

H&S Samuel P. Harbison and Guy L. Steile, C: A Reference Manual, Second Edition, Prentice-Hall, 1987, ISBN 0-13-109802-0. (A third edition has recently been released.)

PCS Mark R. Horton, Portable C Software, Prentice Hall, 1990 ISBN 0-13-868050-7.

K&P Brian W. Kernighan and P.J. Plaugher, The Elements of Programming Style, Second Edition, Mcgraw-Hill, 1987, ISBN 0-07-034207-5

K&B I Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall, 1987, ISBN 0-13-110163-3

K&B II Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Second Edition, Prentice Hall, 1988, ISBN 0-13-110362-8, 0-13-110370-9

CT&P Andrew Koenig, C Traps and Pitfalls, Addison-Wesley, 1989, ISBN 0-2-1-117928-8.